

Wii U Graphics Primer

Textures

2011/11/30

Version 0.1

**The content of this document is highly confidential
and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and/or its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Introduction	5
2	Textures	6
2.1	Texture Count.....	6
2.2	Texture Size	6
2.3	Using Textures	6
2.4	Texture Formats	8
2.4.1	Compressed Formats.....	8
2.4.2	Uncompressed Formats	10
2.5	Comparison of LDR Formats	13
2.6	Comparison of Texture Formats by Data Size	16
2.7	How to Choose a Texture Format	17
2.7.1	How to Choose an LDR Format	17
2.7.2	How to Choose a Format for Normal Mapping.....	18
2.8	Mapping Methods	19
2.8.1	UV Mapping.....	19
2.8.2	Projective Mapping	21
2.8.3	Cube Environment Mapping.....	22
2.9	Methods for Repeating Textures.....	24
2.10	Mipmaps.....	26
2.10.1	Mipmap Levels	27
2.10.2	LOD Bias	28
2.11	Texture Filters	29
2.12	Channel Selection.....	32
2.13	Texture Dimensions	32
2.13.1	1D Textures	32
2.13.2	2D Textures	33
2.13.3	3D Textures	33
	Revision History	35

Tables

Table 2-1 Texture Formats	8
Table 2-2 Types of Compressed Formats.....	8
Table 2-3 Images of Compressed Formats (BC1 to BC3)	9
Table 2-4 Images of Compressed Formats (BC4 and BC5)	10

Table 2-5 Types of Uncompressed Formats.....	10
Table 2-6 Images of Uncompressed Formats (Two or Fewer Channels).....	11
Table 2-7 Images of Uncompressed Formats	11
Table 2-8 Types of HDR Formats	12
Table 2-9 RGB Images in LDR Formats.....	13
Table 2-10 Alpha Images in LDR Formats	14
Table 2-11 Main Texture Formats Used for Normal Mapping.....	18
Table 2-12 Methods for Repeating a Texture	24
Table 2-13 Horizontal/Vertical Filtering Methods.....	29
Table 2-14 Filtering Methods Between Mipmap Levels.....	29
Table 2-15 Texture Filtering With Mipmaps	30
Table 2-16 Anisotropic Filtering	31

Figures

Figure 2-1 Texture Size	6
Figure 2-2 Sample Texture Usage (Normal Mapping).....	7
Figure 2-3 Sample Texture Usage (Specular Mapping)	7
Figure 2-4 Sample Texture Usage (Occlusion Mapping)	7
Figure 2-5 HDR Texture.....	12
Figure 2-6 Data Size of Different Texture Formats (for a 1024x1024 Texture)	16
Figure 2-7 How to Choose Between Compressed and Uncompressed Formats	17
Figure 2-8 Texture Coordinate Space	19
Figure 2-9 UV Mapping	20
Figure 2-10 Projective Mapping	21
Figure 2-11 Cube Environment Mapping.....	22
Figure 2-12 Cube Map Textures	23
Figure 2-13 Methods for Repeating a Texture.....	25
Figure 2-14 Sample Application of Mipmaps.....	26
Figure 2-15 Example of Using Mipmaps to Remove Moiré Patterns	26
Figure 2-16 Mipmap Levels.....	27
Figure 2-17 LOD Bias.....	28
Figure 2-18 Horizontal/Vertical Point Sampling and Bilinear Filtering.....	29
Figure 2-19 Sample Channel Selection	32
Figure 2-20 A 1D Texture.....	32
Figure 2-21 A 2D Texture.....	33
Figure 2-22 A 3D Texture.....	33
Figure 2-23 Example of Applying a 3D Texture.....	34

1 Introduction

The purpose of this document is to explain features and restrictions that affect textures on the Wii U. A basic understanding of 3D graphics is a prerequisite for fully understanding this document.

The document does not go into detailed procedures for 3D programming or shader programming. Descriptions of some features have been omitted, along with descriptions of some program-related restrictions and other information. For more details on each of these features, see the other documentation.

2 Textures

2.1 Texture Count

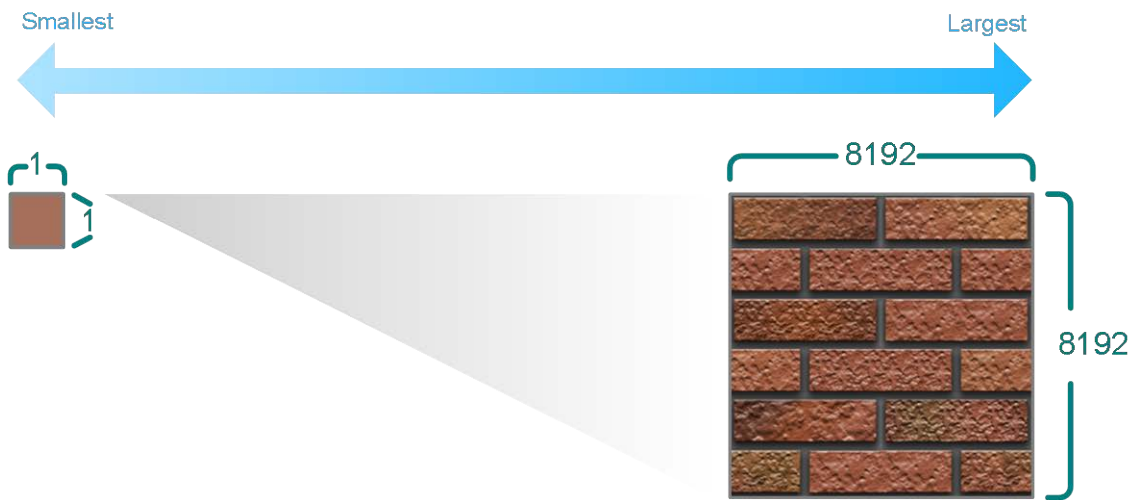
Up to 16 textures can be used simultaneously on the Wii U.

Note: Future updates to the Wii U SDK may allow even more textures to be used simultaneously.

2.2 Texture Size

Each side of a texture can be between 1 and 8192 texels long.

Figure 2-1 Texture Size

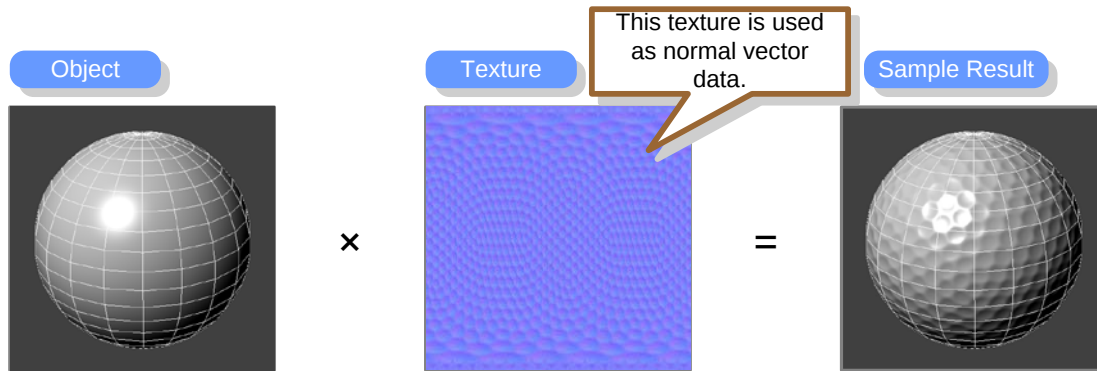
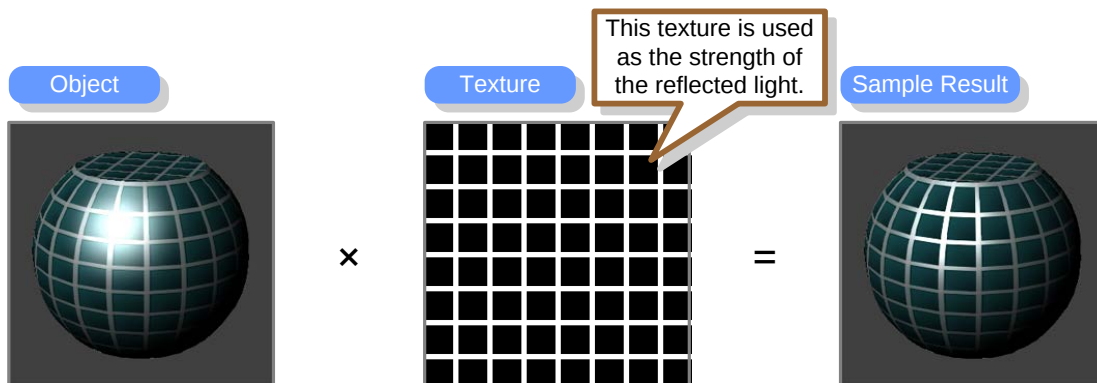
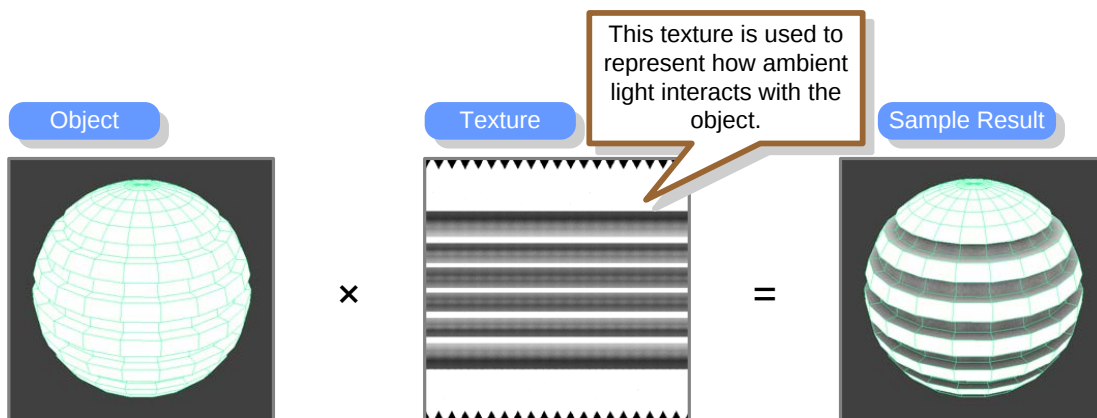


Note: The cost of processing a texture increases with its size.

2.3 Using Textures

Textures may be used as color information, and for various other purposes. Here are some examples:

- Normal mapping (see Figure 2-2)
This uses textures to represent normal vector data.
- Specular mapping (see Figure 2-3)
This uses textures to represent the strength of reflections.
- Occlusion mapping (see Figure 2-4)
This uses textures to represent how ambient light hits an object.

Figure 2-2 Sample Texture Usage (Normal Mapping)**Figure 2-3 Sample Texture Usage (Specular Mapping)****Figure 2-4 Sample Texture Usage (Occlusion Mapping)**

2.4 Texture Formats

Texture formats are divided into the following two types.

- Compressed formats
These formats compress the data of some or all channels.
- Uncompressed formats
These formats do not compress the data of any channel.

Uncompressed formats are further divided into the following two types, depending on the number of color levels the format can display.

- Low Dynamic Range (LDR) formats
These formats use 8 bits or less for each channel. These have color levels that can be displayed on ordinary monitors.
- High Dynamic Range (HDR) formats
These formats use more than 8 bits for each channel.

HDR formats cannot be compressed.

Table 2-1 Texture Formats

	LDR Formats	HDR Formats
Are compressed formats available?	Yes	No
Are uncompressed formats available?	Yes	Yes

2.4.1 Compressed Formats

Texel information is compressed when it is saved in these formats.

The compression method that is used affects both the texture's compression ratio and its suitability for particular uses. Because these formats use lossy compression, after compression it is not always possible to faithfully reproduce the original image.

There are five types of compressed formats, as shown in Table 2-2. Only LDR formats are provided.

Table 2-2 Types of Compressed Formats

Format	Internal Structure (Bits)	Bits/Texel (After compression)
BC1 (without alpha)	R5 G6 B5	4
BC1 (with alpha)	R5 G6 B5 A1	4
BC2	R5 G6 B5 A4	8
BC3	R5 G6 B5 A8	8
BC4	R8	4
BC5	R8 G8	8

Table 2-3 Images of Compressed Formats (BC1 to BC3)

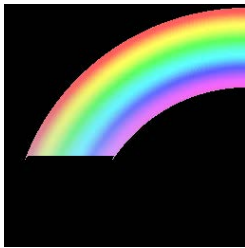
Format	RGB Image	Alpha Image	Description
Original image (R8G8B8A8)			This is the original, uncompressed image. Each channel has 8 bits (256 color levels).
BC1 (without alpha)		N/A	Color depth is reduced to R5 G6 B5 and then each texel is compressed into 4 bits.
BC1 (with alpha)			Color depth is reduced to R5 G6 B5 A1 and then each texel is compressed into 4 bits.
BC2			Color depth is reduced to R5 G6 B5 A4 and then each texel is compressed into 8 bits.
BC3			Color depth is reduced to R5 G6 B5 A8 and then each texel is compressed into 8 bits.

Table 2-4 Images of Compressed Formats (BC4 and BC5)

Format	Image for the First Channel	Image for the Second Channel	Description
Original image (R8G8)			This is the original, uncompressed image. Each channel has 8 bits (256 color levels).
BC4		N/A	R8 is compressed into 4 bits.
BC5			R8 G8 is compressed into 8 bits.

2.4.2 Uncompressed Formats

These formats save uncompressed texel information. There are both LDR and HDR formats.

2.4.2.1 LDR Formats

These formats use 8 bits or less for each channel. They have color levels that can be displayed on ordinary monitors.

There are six formats, differentiated by their internal structure, as shown in Table 2-5.

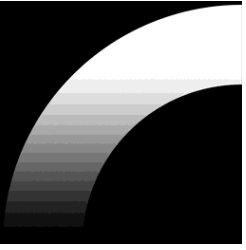
Table 2-5 Types of Uncompressed Formats

Format	Internal Structure (Bits)	Bits/Texel
R8	R8	8
R8G8	R8 G8	16
R5G6B5	R5 G6 B5	16
R4G4B4A4	R4 G4 B4 A4	16
R5G5B5A1	R5 G5 B5 A1	16
R8G8B8A8	R8 G8 B8 A8	32

Table 2-6 Images of Uncompressed Formats (Two or Fewer Channels)

Format	Image for the First Channel	Image for the Second Channel	Description
R8		N/A	A single 8-bit channel (with 256 color levels).
R8G8			Two 8-bit channels (with 256 color levels each).

Table 2-7 Images of Uncompressed Formats

Format	RGB Image	Alpha Image	Description
R5G6B5		N/A	Two 5-bit channels (with 32 color levels each) and one 6-bit channel (with 64 color levels).
R4G4B4A4			Four 4-bit channels (with 16 color levels each).
R5G5B5A1			Three 5-bit channels (with 32 color levels each) and one 1-bit channel.

Format	RGB Image	Alpha Image	Description
R8G8B8A8			Four 8-bit channels (with 256 color levels each).

2.4.2.2 HDR Formats

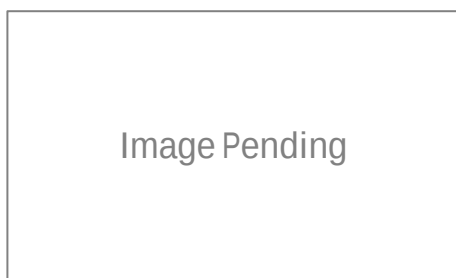
These texture formats are mainly used for High Dynamic Range (HDR) images. They can reproduce more detailed shadows and highlights than formats with 8 bits or less per channel.

There are nine formats differentiated by their internal structure, as shown in the following table.

Table 2-8 Types of HDR Formats

Format	Internal Structure (Bits)	Bits/Texel
R16	R16	16
R32	R32	32
R16G16	R16 G16	32
R32G32	R32 G32	64
R11G11B10	R11 G11 B10	32
R10G10B10A2	R10 G10 B10 A2	32
R16G16B16A16	R16 G16 B16 A16	64
R32G32B32A32	R32 G32 B32 A32	128

Figure 2-5 HDR Texture



2.5 Comparison of LDR Formats

The following tables show the same images in each of the LDR formats.

Table 2-9 RGB Images in LDR Formats

Format	Images	
R5G6B5		
R4G4B4A4		
R5G5B5A1		
R8G8B8A8		
BC1 (without alpha) BC2 BC3		

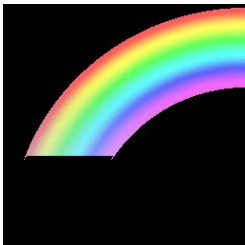
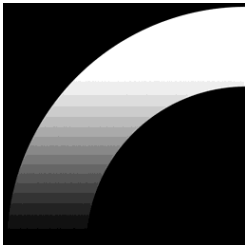
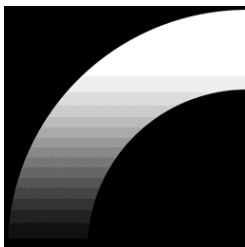
Format	Images
BC1 (with alpha)	

Table 2-10 Alpha Images in LDR Formats

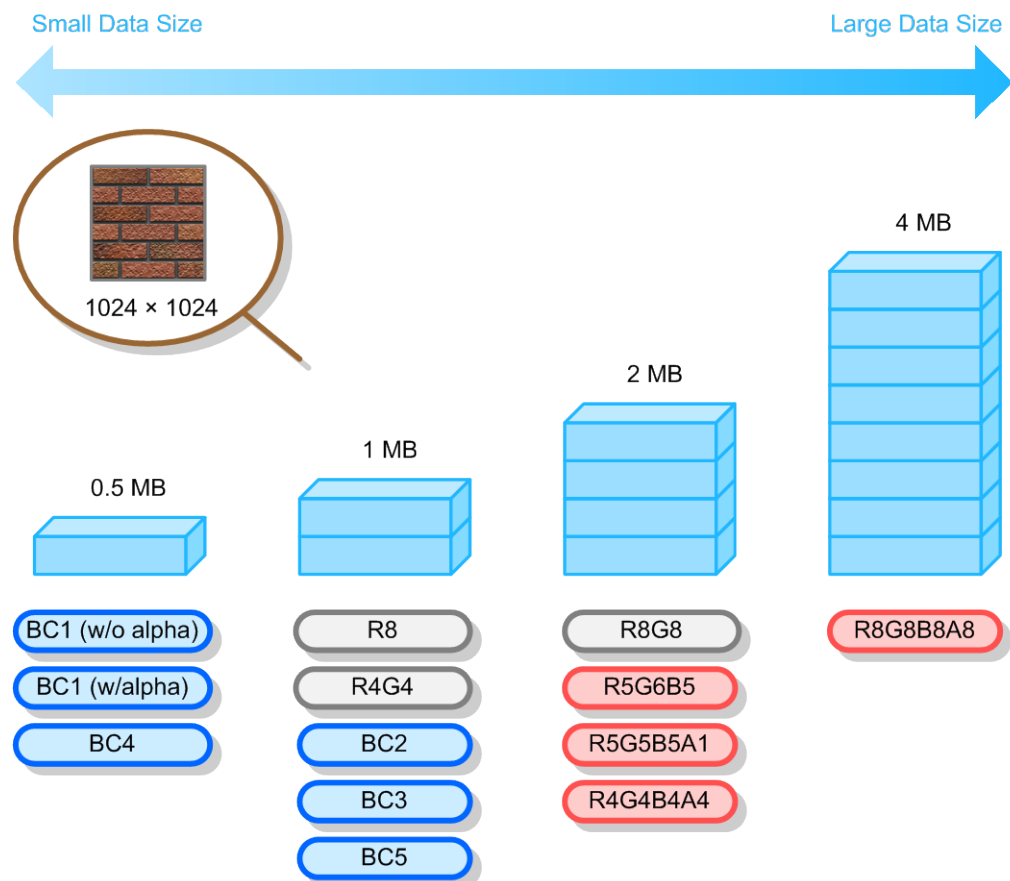
Format	Images
R5G5B5A1	
R4G4B4A4	
R8G8B8A8	
BC1 (with alpha)	

Format	Images
BC2	
BC3	

2.6 Comparison of Texture Formats by Data Size

Texture data size is determined by the texture size (width by height), and the number of bits per texel in the texture format of that texture. For example, the following figure compares the memory usage of each format for a 1024x1024 texture. Memory is used much more efficiently when you choose a texture format compatible with the intended use of the texture.

Figure 2-6 Data Size of Different Texture Formats (for a 1024x1024 Texture)

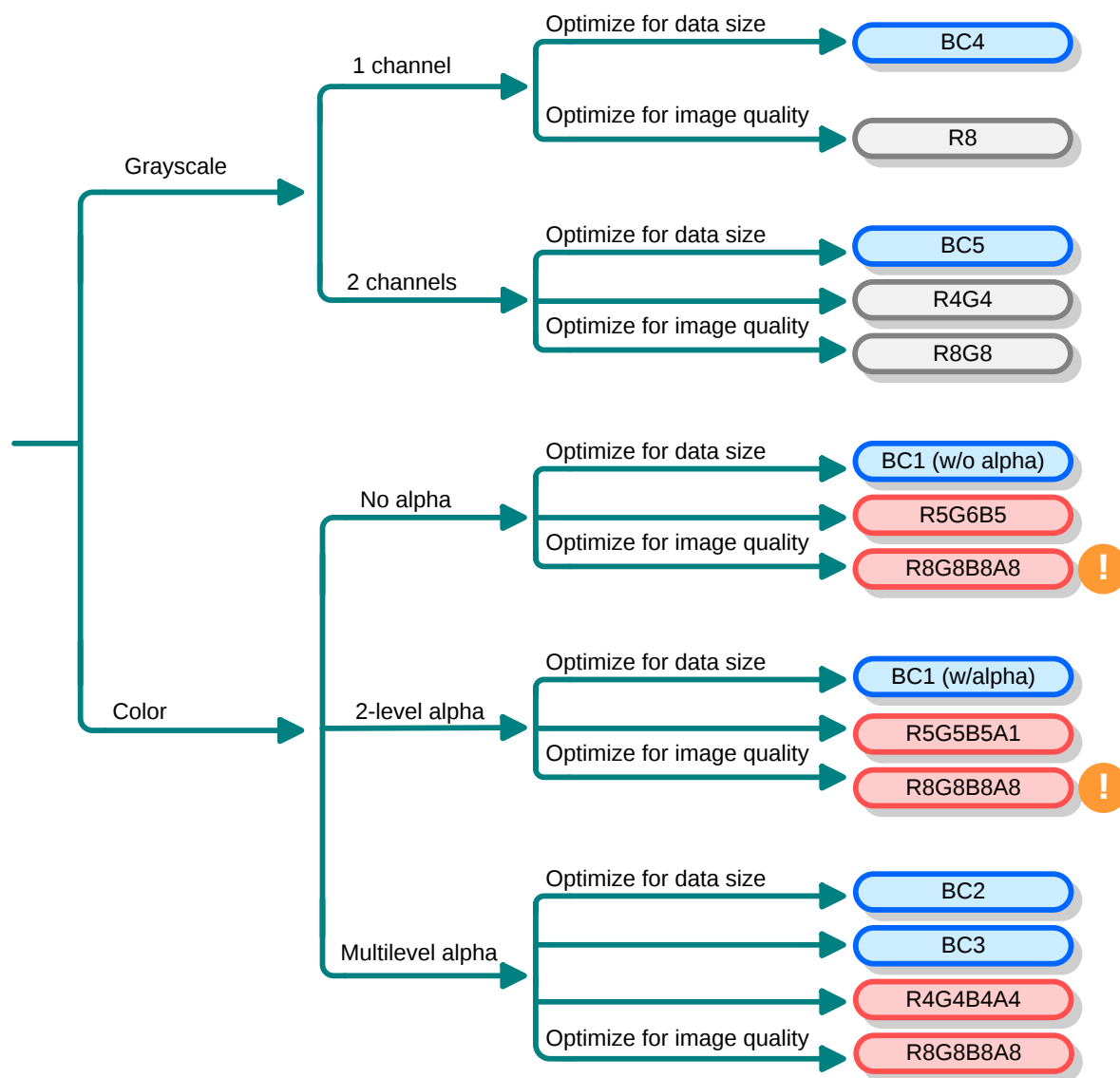


2.7 How to Choose a Texture Format

2.7.1 How to Choose an LDR Format

The following figure shows the process of choosing an LDR format.

Figure 2-7 How to Choose Between Compressed and Uncompressed Formats



If the RGB channels must each have 8 bits, use R8G8B8A8 even if you don't use any or all of the bits in the alpha channel.

2.7.2 How to Choose a Format for Normal Mapping

The following table shows the main formats used to apply a texture as normal vector data for normal mapping. Take into account the properties of each format to choose the one best suited for your usage.

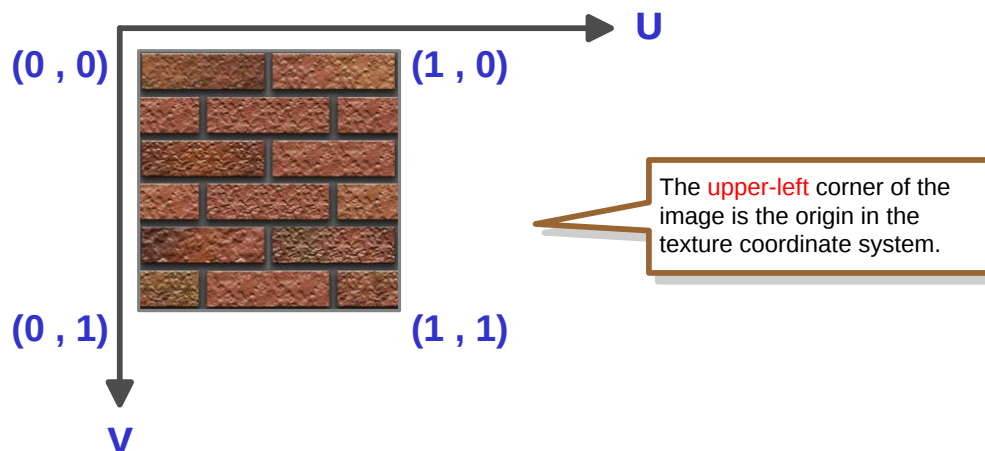
Table 2-11 Main Texture Formats Used for Normal Mapping

Format	Image With Texture Applied	Data Size	Precision of the Normal Vector Data	Comments
BC5	Image Pending	Small	Somewhat high	This format is used the most. It has a good balance between data size and precision.
BC3	Image Pending	Small	Low	The alpha channel can be used to carry information that requires precision.
BC1 (w/o alpha)	Image Pending	Very small	Low	Use this when you need to reduce the data size as much as possible.
R8G8	Image Pending	Large	High	Use this when you have plenty of space for data.
R8G8B8A8	Image Pending	Very large	High	The alpha channel can be used to carry information other than normal vector data.

2.8 Mapping Methods

The upper-left corner of an image is the origin in the texture coordinate system. The positive U axis extends to the right and the positive V axis extends down. Regardless of its actual dimensions, a texture is mapped so that it fits precisely between 0.0 and 1.0 in both the U and V directions.

Figure 2-8 Texture Coordinate Space



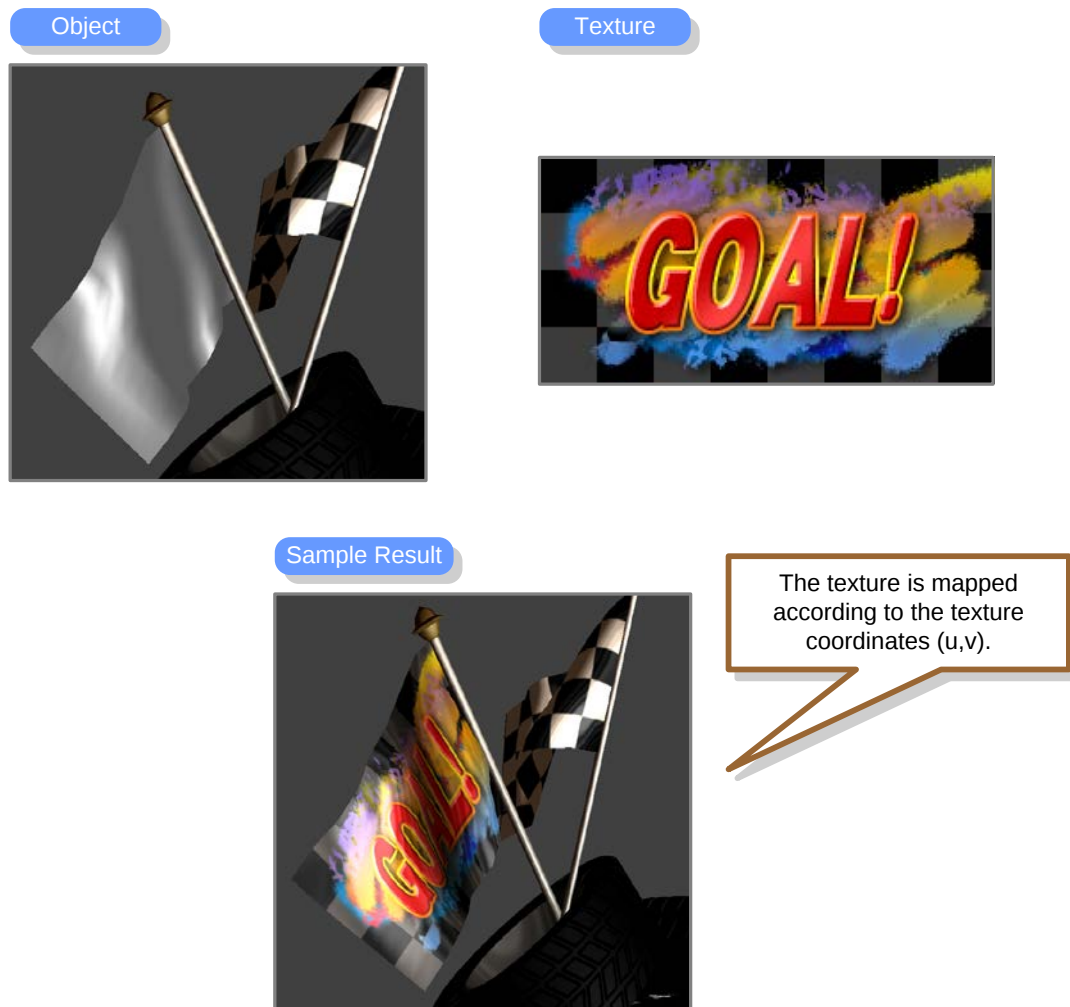
You can use the following texture mappings.

- UV mapping.
- Projective mapping.
- Cube environment mapping.

2.8.1 UV Mapping

This mapping method maps a texture to texture coordinates that are input as vertex attributes. It is the most common way to map textures.

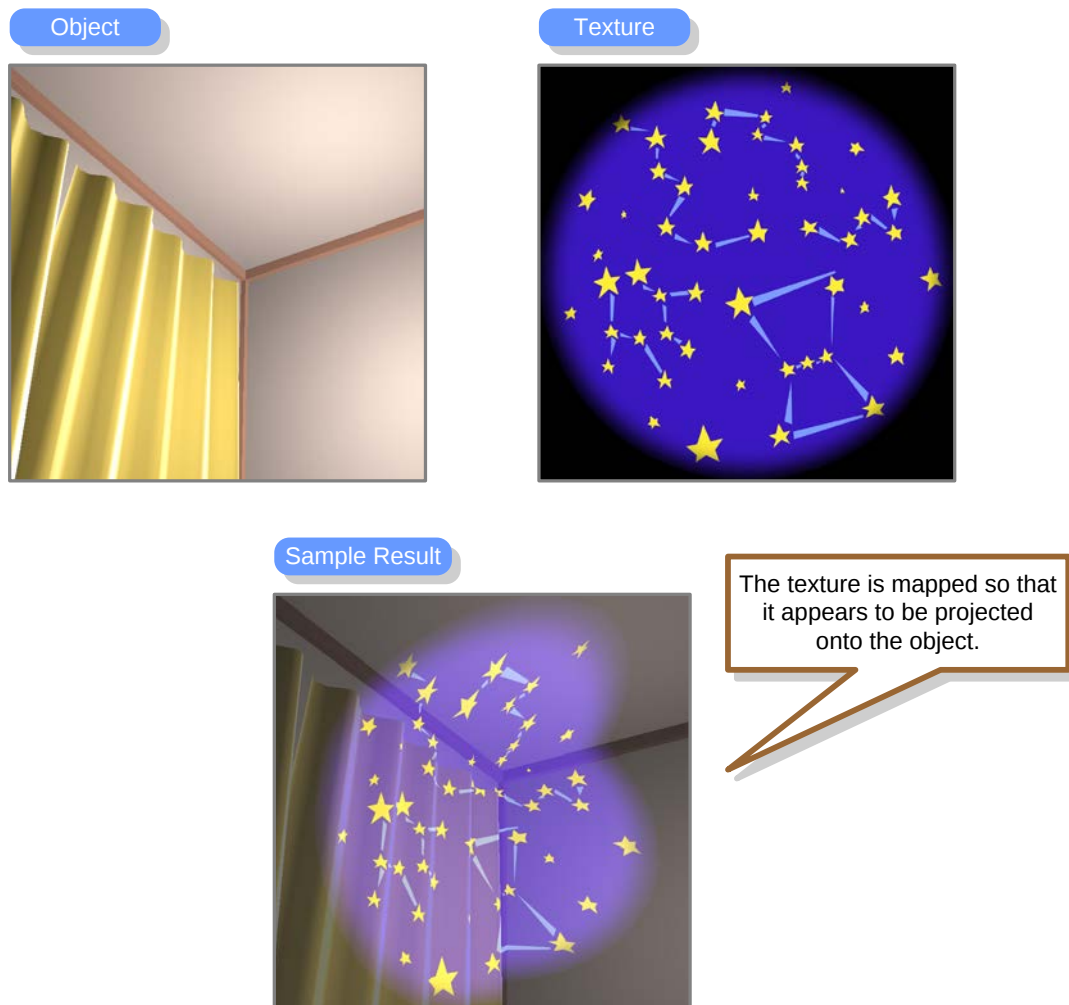
Figure 2-9 UV Mapping



2.8.2 Projective Mapping

This method maps a texture by generating texture coordinates from an object's vertex coordinates, and thus, allows you to depict spotlights, shadows, images projected as if by a slide projector, and so forth.

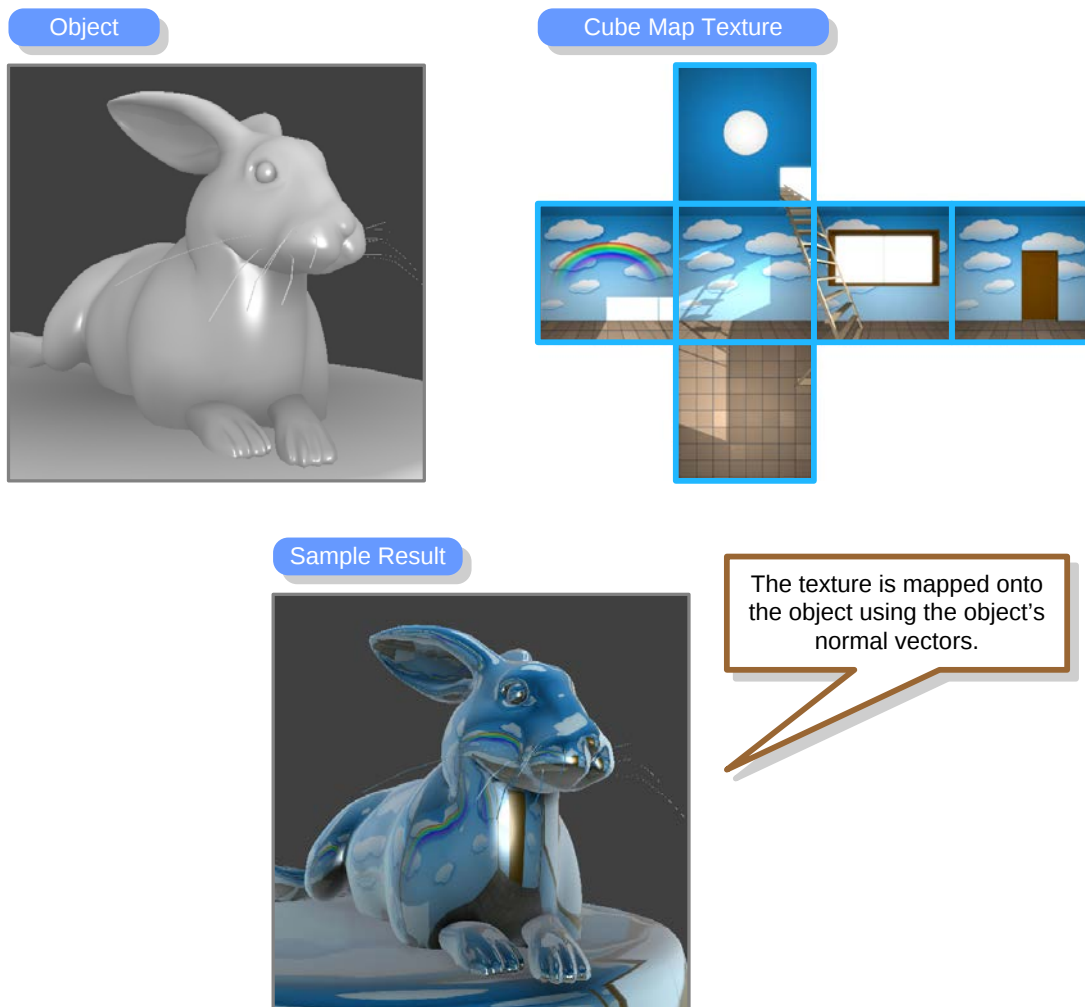
Figure 2-10 Projective Mapping



2.8.3 Cube Environment Mapping

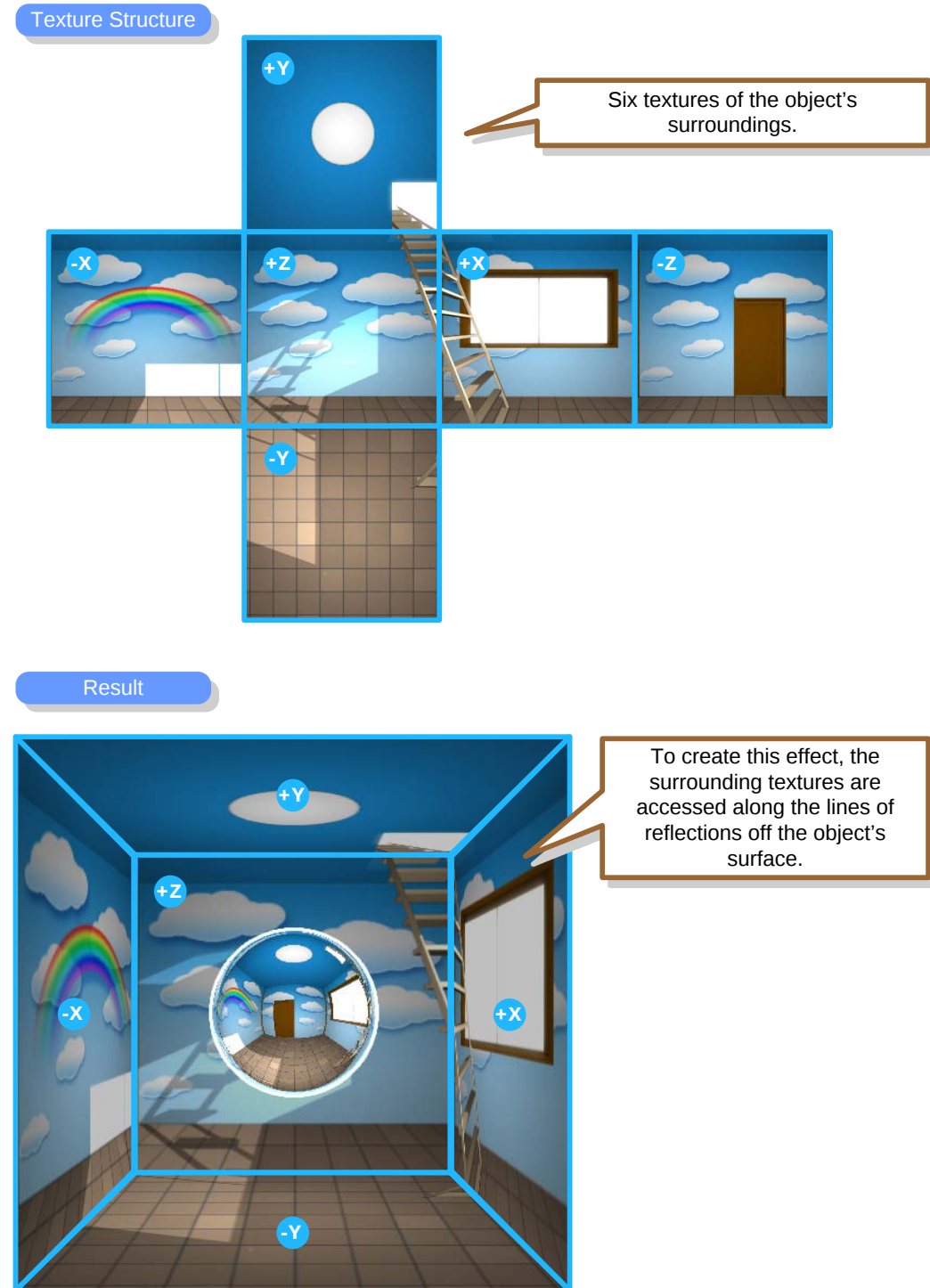
This method maps a texture by generating texture coordinates from an object's normal and view vectors. By preparing a cube map texture of a scene, you can reflect the scene onto an object placed within the scene.

Figure 2-11 Cube Environment Mapping



A cube map texture actually comprises six textures, one for each direction around an object. In each direction the textures must be square. Placed side-by-side, the textures look like an unfolded cube.

Figure 2-12 Cube Map Textures



2.9 Methods for Repeating Textures

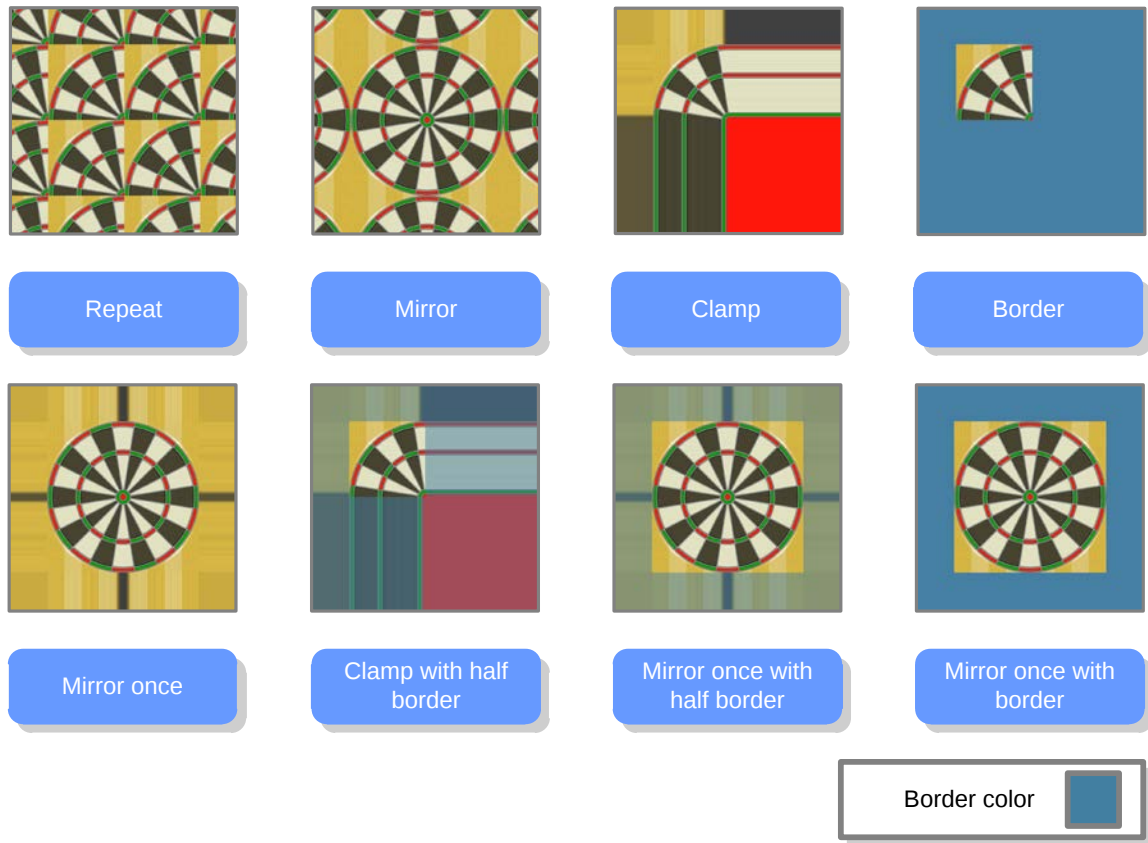
You can choose from the following eight methods of mapping repeated textures. A texture can be repeated horizontally, vertically, or in both directions.

Table 2-12 Methods for Repeating a Texture

Method	Description
Repeat	Repeats the texture.
Mirror	Flips and repeats the texture.
Clamp	Extends the colors at the edges of the texture.
Border	Fills the region around the texture with a specified border color.
Mirror once	Repeats the texture just three times: once flipped horizontally, once flipped vertically, and once flipped both horizontally and vertically, then extends the colors at the edges.
Clamp with half border	Extends the colors at the edges of the texture, blended with the border color.
Mirror once with half border	Repeats the texture just three times: once flipped horizontally, once flipped vertically, and once flipped both horizontally and vertically, then extends the colors at the edges, blended with the border color.
Mirror once with border	Repeats the texture just three times: once flipped horizontally, once flipped vertically, and once flipped both horizontally and vertically, then fills the region around the texture with the specified border color.

The following figure gives samples of the different methods for repeating textures, shown repeated both horizontally and vertically.

Figure 2-13 Methods for Repeating a Texture



2.10 Mipmaps

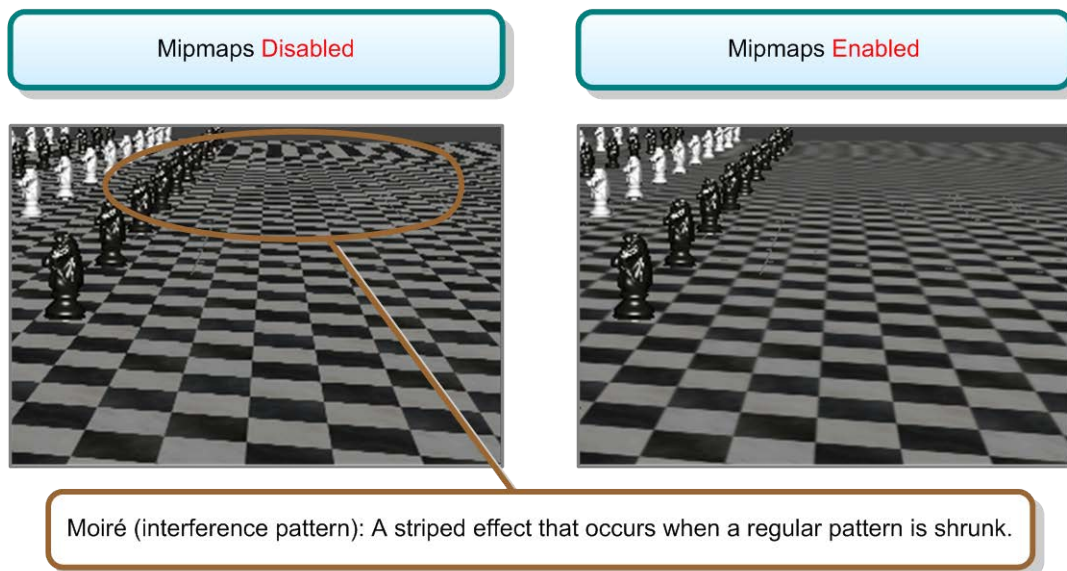
Mipmapping is a technique in which low-resolution textures are prepared in advance, and then applied during the process of displaying minified textures. *Mipmaps* are the textures of various resolutions used for mipmapping.

Figure 2-14 Sample Application of Mipmaps



You can use mipmaps to reduce the processing load. Mipmaps can also suppress moiré patterns and other unnatural effects.

Figure 2-15 Example of Using Mipmaps to Remove Moiré Patterns



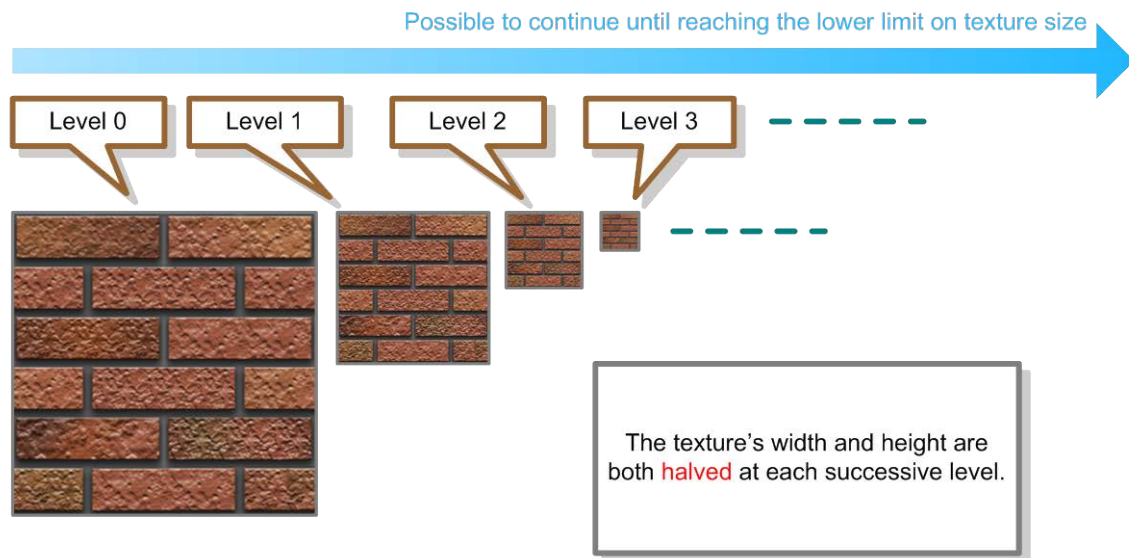
Note: You may also be able to improve performance by using mipmaps when objects are not a fixed distance away from the camera. Because of the general possibility of performance improvement, Nintendo recommends the use of mipmaps in all situations.

2.10.1 Mipmap Levels

Mipmap levels are the successive stages of textures used for mipmaps. Given level 0 as the size of the original texture, each successive level halves the width and height of the texture.

You can use up to a maximum of 14 mipmap levels, which is the theoretical maximum number of levels it takes for both the width and height of the mipmap texture to reach the smallest allowable size.

Figure 2-16 Mipmap Levels



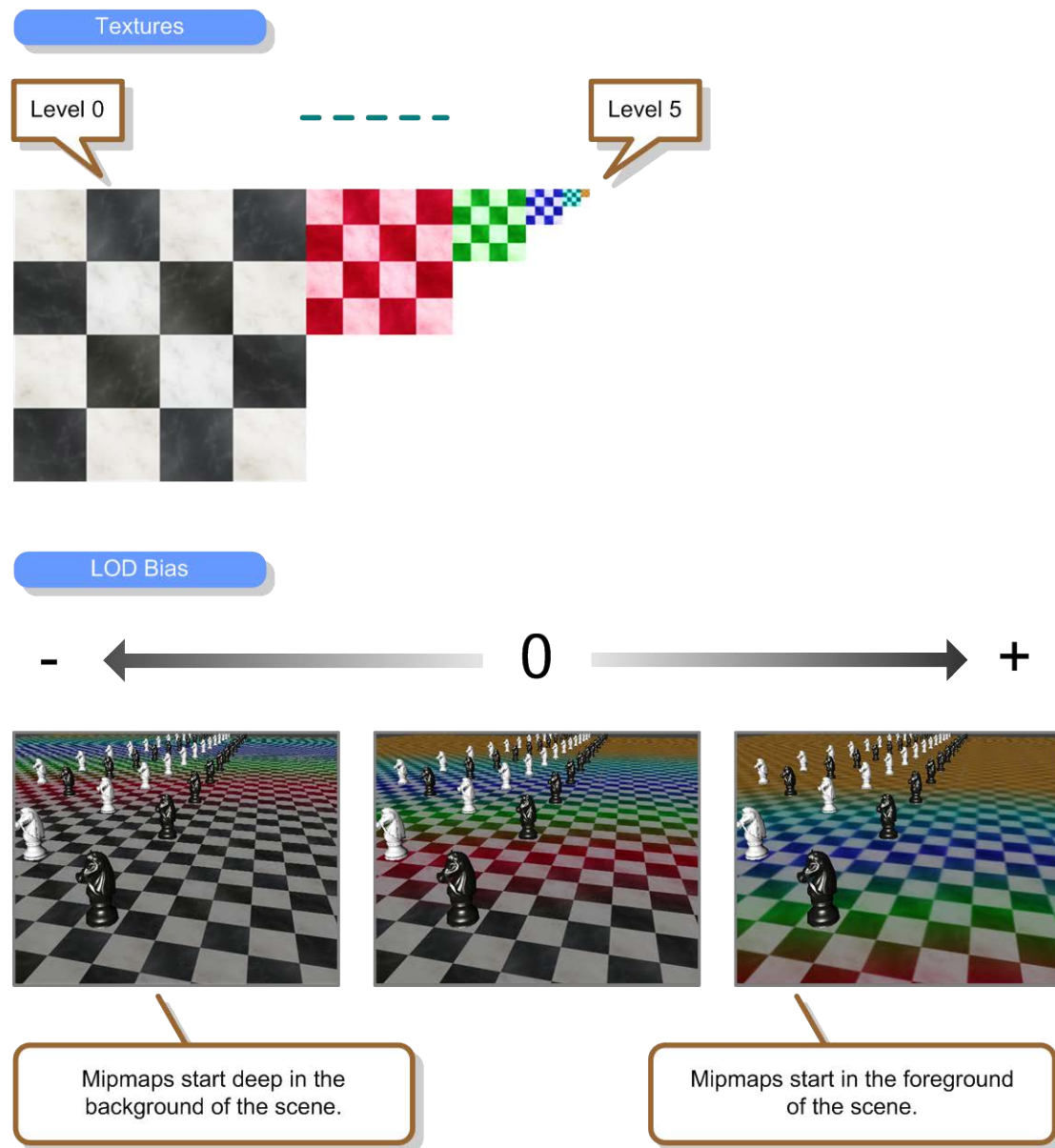
Note: The maximum texture size is 8192x8192. If you halve this 13 times, you will end up with the minimum texture size of 1x1. You can, therefore, have a maximum of 14 mipmap levels.

2.10.2 LOD Bias

Mipmapped textures may look overly blurred where low-resolution mipmap levels are accessed.

LOD bias is a feature that adjusts where mipmaps begin to be applied. The following figure uses a different texture color for each mipmap level to make the levels easier to see.

Figure 2-17 LOD Bias



2.11 Texture Filters

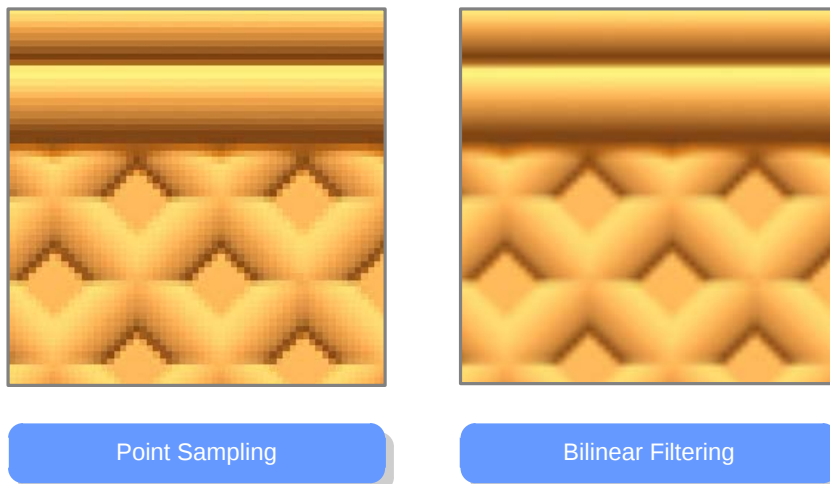
When mapping any single texture, you can specify one filtering method to apply between mipmap levels and a different method to apply horizontally and vertically within the texture.

The following table shows the two methods for filtering horizontally and vertically, within a single texture. You can, separately, specify which filtering method to use during texture magnification, and which to use during texture minification.

Table 2-13 Horizontal/Vertical Filtering Methods

Filtering Method	Description
Point sampling	This method determines a fragment's color by accessing its corresponding texel precisely. Each texel is distinctly displayed.
Bilinear filtering	This method determines a fragment's color by interpolating the four adjacent texel colors. This displays smooth textures.

Figure 2-18 Horizontal/Vertical Point Sampling and Bilinear Filtering



You can also specify a texture filter to apply between mipmap levels, where mipmaps are used to display a minified texture.

The following table shows the two methods for filtering between mipmap levels.

Table 2-14 Filtering Methods Between Mipmap Levels

Filtering Method	Description
Point sampling	This method uses the texture at the mipmap level that is closest to an object's size.
Linear filtering	This method interpolates between the textures for two mipmap levels that are close to an object's size.

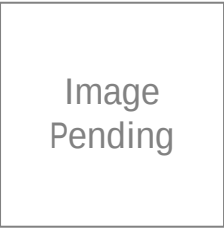
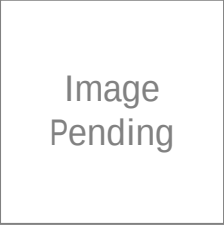
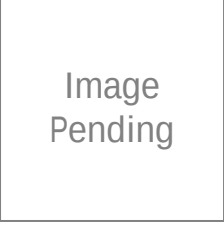
Table 2-15 Texture Filtering With Mipmaps

		Horizontal/Vertical Filtering Methods Within the Same Mipmap Level	
		Point Sampling	Bilinear Filtering
Filtering Method Between Mipmap Levels	Point Sampling		
	Linear Filtering		

The Wii U can perform anisotropic filtering at the same time as the other filtering methods explained so far. Anisotropic filtering makes textures appear smooth, even on objects that are viewed from oblique angles.

You can choose from the five levels of anisotropic filtering quality shown in the following table. 1:1 filtering (no anisotropic filtering) represents the lowest quality, and 16:1 filtering represents the highest. Computational overhead increases with quality.

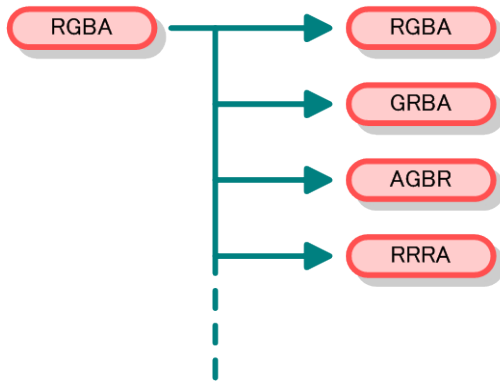
Table 2-16 Anisotropic Filtering

Quality	Image
1:1 (no anisotropic filtering)	
2:1	
4:1	
8:1	
16:1	

2.12 Channel Selection

Any of a texture's RGBA channels may be chosen as the colors to actually use for rendering. You can switch the red and alpha channels, or copy the red channel to a different channel, for instance.

Figure 2-19 Sample Channel Selection



2.13 Texture Dimensions

The Wii U can handle the following three types of textures, each of which has a different number of dimensions. Each type of texture requires a different number of coordinates to access a texel within that texture.

- 1D textures.
- 2D textures.
- 3D textures.

2.13.1 1D Textures

A 1D texture only has information in a single direction. Its texels can be bound to a single texture coordinate. In the following figure, each texel is accessed by its U coordinate.

Figure 2-20 A 1D Texture

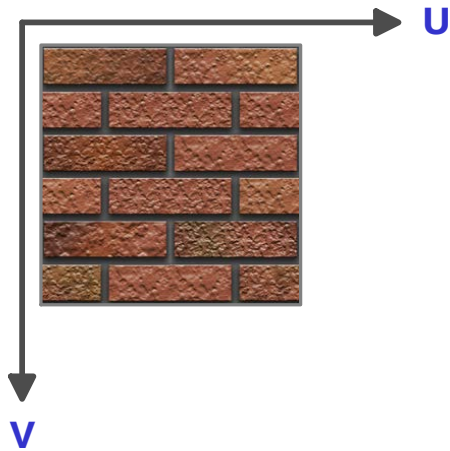


You can use a 1D texture as a table to convert U coordinates into color information or for some other purpose, besides mapping an image to an object.

2.13.2 2D Textures

A 2D texture has information in two directions. This is the most common type of texture. Its texels can be bound to two texture coordinates. In the following figure, each texel is accessed by its U and V coordinates.

Figure 2-21 A 2D Texture

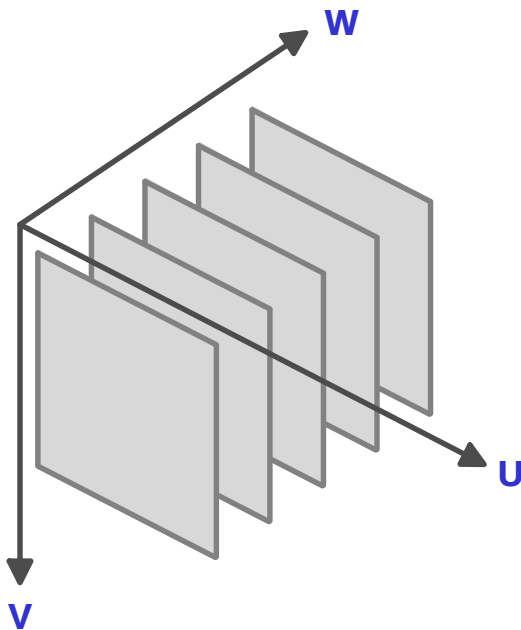


Note: All other sections in this document use 2D textures for their examples.

2.13.3 3D Textures

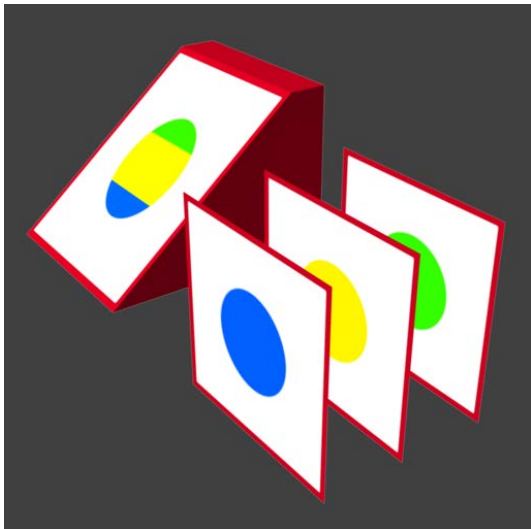
A 3D texture has information in three directions. Its texels can be bound to three texture coordinates. In the following figure, each texel is accessed by its U, V, and W coordinates.

Figure 2-22 A 3D Texture



By applying a 3D texture to an object with the texture's axes corresponding to the object's width, height, and depth, you can depict an image of the inside of the object.

Figure 2-23 Example of Applying a 3D Texture



A 3D texture can range from 1 to 8192 texels deep. You can use mipmapped 3D textures and filter 3D textures in the depth direction.

Revision History

Version	Revision Date	Description
0.1	2011/11/30	Initial version.

All company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2012 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.